This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently amended) A system for compression, the system comprising:

a memory device that stores a b-tree data structure comprising a plurality of compressed and uncompressed normalized index keys, each normalized index key generated by normalizing a plurality of column values that constitute an index key, the normalized index keys stored in sorted order, with no gaps between the stored normalized index keys, and stores a plurality of slots with no gaps between the stored slots, wherein the memory device stores the plurality of compressed and uncompressed normalized index keys starting after a header and the plurality of normalized index keys grows towards an end of the memory device as additional index keys are added; and,

a processor that compresses the stored normalized keys on [[the]] a memory page by:

(a) determining if a first normalized index key in [[a]] the memory device should be compressed;

(b) comparing the first normalized index key with a second normalized index key preceding the first normalized index key in the memory device;

(c) generating a common byte length between the first normalized index key and the second normalized index key consisting of [[the]] a number of bytes in [[the]] a common prefix between the first normalized index key and the second normalized index key;

(d) replacing the first normalized index key in the memory page with the generated common byte length followed by the bytes from the first normalized index key that were not in the common prefix between the first normalized index key and the second normalized index key;

(e) shifting the normalized index keys following the first normalized index key to fill any empty memory space resulting from compressing the first normalized index key and updating [[the]] memory offsets contained in the slots corresponding to the shifted normalized index keys; and

(f) updating [[the]] an indicator in the slot corresponding to the first normalized index key to reflect that the key is now compressed,

wherein each slot corresponds to a normalized index key in the memory page and comprises a memory offset of the corresponding key and an indicator indicating if the

corresponding normalized index key is compressed, wherein the processor compresses the stored normalized index keys before a memory page split.

2-5. (Cancelled)

6. (Previously Presented) The system of claim 1, further comprising the processor repeating steps (a) – (f) for each normalized index key in the memory device.

7. (Currently amended) The system of claim 1, wherein the processor determining if a first normalized index key should be compressed comprises:

examining [[an]] the indicator in the slot corresponding to the first normalized index key to determine if the first normalized key is already compressed and not compressing a key that has already been compressed; and

determining if the first normalized index key has a preceding index key on the memory device and not compressing a key that does not have a preceding index key on [[a]] the memory device.

8. (Cancelled)

9. (Currently amended) A method for compressing a b-tree data structure, the method comprising the following steps:

storing a plurality of compressed and uncompressed normalized index keys of [[a]] the b-tree data structure in sorted order in a memory page with no gaps between the stored normalized keys, wherein each index key comprises a plurality of columns each having a column value and a column type and is normalized by normalizing each column value using a normalization function selected based on the column type and concatenating the normalized column values;

storing a plurality of slots with no gaps between the stored slots;

storing a header;

compressing by a computer processor the stored normalized index keys on the memory page by

(a) determining if a first normalized index key in [[a]] the memory page should be compressed;

(b) comparing the first normalized index key with a second normalized index key preceding the first normalized index key in the memory page;

(c) generating a common byte length between the first normalized index key and the second normalized index key comprising [[the]] a number of bytes in [[the]] a common prefix between the first normalized index key and the second normalized index key;

(d) replacing the first normalized index key in the memory page with the generated common byte length followed by the bytes from the first normalized index key that were not in the common prefix between the first normalized index key and the second normalized index key;

(e) shifting the normalized index keys following the first normalized index key to fill any empty memory space resulting from compressing the first normalized index key and updating [[the]] memory offsets contained in the slots corresponding to the shifted normalized index keys; and

(f) updating [[the]] an indicator in the slot corresponding to the first normalized index key to reflect that the key is now compressed; and

after compressing [[a]] each stored normalized index key, shifting at least one other stored normalized index key to fill an empty memory space resulting from compressing the stored normalized index key, wherein storing the plurality of slots comprises starting immediately at [[the]] an end of the memory page and growing towards [[the]] a beginning of the memory page as additional slots are added, further wherein each slot corresponds to a normalized index key in the memory page and comprises [[of]] a memory offset of the corresponding key and an indicator indicating if the corresponding normalized index key is compressed,

wherein compressing the stored normalized index keys is performed before a memory page split.

10. (Previously Presented) The method of claim 9, wherein storing the plurality of compressed and uncompressed normalized index keys comprises starting following the

header with the plurality of normalized index keys growing towards the end of the memory page as additional index keys are added.

11-13. (Cancelled)

14. (Previously Presented) The method of claim 1, further comprising repeating steps (a) – (f) for each normalized index key in the memory page.

15. (Currently amended) The method of claim 1, wherein the determining if a first normalized index key should be compressed comprises:

examining [[an]] the indicator in the slot corresponding to the first normalized index key to determine if the first normalized key is already compressed and not compressing a key that has already been compressed; and

determining if the first normalized index key has a preceding index key on the memory page and not compressing a key that does not have a preceding index key on [[a]] the memory page.

16. (Cancelled)

17. (Currently amended) A method for compressing normalized index keys in a b-tree data structure, the method comprising the following steps:

(a) determining by a computer processor if a first normalized index key in a memory page of [[a]] the b-tree data structure should be compressed, wherein the first normalized index key is generated by normalizing a plurality of first column values that constitute a first index key and concatenating the normalized first column values, wherein determining if a first normalized index key should be compressed comprises:

examining an indicator in the slot corresponding to the first normalized index key to determine if the first normalized key is already compressed and not compressing a key that has already been compressed; and

determining if the first normalized index key has a preceding index key on the memory page and not compressing a key that does not have a preceding index key on [[a]] the memory page;

(b) comparing by a computer processor the first normalized index key with a second normalized index key preceding the first normalized index key in the memory page, wherein the second normalized index key is generated by normalizing a plurality of second column values that constitute a second index key and concatenating the normalized second column values;

(c) generating by a computer processor a common byte length between the first normalized index key and the second normalized index key comprising [[the]] a number of bytes in [[the]] a common prefix between the first normalized index key and the second normalized index key;

(d) replacing by a computer processor the first normalized index key in the memory page with the generated common byte length followed by the bytes from the first normalized index key that were not in the common prefix between the first normalized index key and the second normalized index key;

(e) shifting by a computer processor the normalized index keys following the first normalized index key to fill any empty memory space resulting from compressing the first normalized index key and updating [[the]] memory offsets contained in the slots corresponding to the shifted normalized index keys; and

(f) updating by a computer processor the indicator in the slot corresponding to the first normalized index key to reflect that the key is now compressed.

18. (Previously Presented) The method of claim 17, further comprising repeating steps (a) – (f) for each normalized index key in the memory page.

19-23. (Cancelled)